APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

ADAPTIVE PAGE MANAGEMENT

Inventor: James M. Dodd
Docket Number: P16947

Prepared by: Jeffrey B. Huter
Patent Attorney

Intel Corporation
5000 W. Chandler Blvd., CH6-404
Chandler, AZ 85226-3699
Phone: (480) 554-4198
Facsimile: (480) 554-7738

"Express Mail" Label Number: EL 962027935 US

ADAPTIVE PAGE MANAGEMENT

BACKGROUND

[0001] Computing devices typically comprise a processor, memory, and a memory controller to provide the processor as well as other components of the computing device with access to the memory. The performance of such computing devices is strongly influenced by the "memory read latency" and "memory write latency" of the computing device. In general, the "memory read latency" is the length of time between when the processor requests the memory controller to retrieve data from the memory and when the memory controller provides the processor with the requested data. Similarly, the "memory write latency" is generally the length of time between when the processor requests the memory controller to write data to the memory and when the memory controller indicates to the processor that the data has been or will be written to the memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The invention described herein is illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. For example, the dimensions of some elements may be exaggerated relative to other elements for clarity. Further, where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0003] FIG. 1 illustrates an embodiment of a computing device.

[0004] FIG. 2 illustrates another embodiment of a computing device.

[0005] FIG. 3 illustrates an embodiment of a hierarchal memory arrangement of the memory depicted in FIGS. 1 and 2.

[0006] FIG. 4 illustrates an embodiment of the memory controllers depicted in FIGS. 1 and 2.

[0007] FIG. 5 illustrates an embodiment of a counter used in one embodiment of adaptive page close logic depicted in FIG. 4.

[0008] FIG. 6 illustrates an embodiment of a method that may be used by the memory controllers of FIGS. 1 and 2 to generate page close messages.

DETAILED DESCRIPTION

[0009] The following description describes techniques for managing pages of a memory device. In the following description, numerous specific details such as logic implementations, opcodes, means to specify operands, resource partitioning/sharing/duplication implementations, types and interrelationships of system components, and logic partitioning/integration choices are set forth in order to provide a more thorough understanding of the present invention. It will be appreciated, however, by one skilled in the art that the invention may be practiced without such specific details. In other instances, control structures, gate level circuits and full software instruction sequences have not been shown in detail in order not to obscure the invention. Those of ordinary skill in the art, with the included descriptions, will be able to implement appropriate functionality without undue experimentation.

[0010] References in the specification to "one embodiment", "an embodiment", "an example embodiment", etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may not necessarily include the particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described.

[0011] Embodiments of the invention may be implemented in hardware, firmware, software, or any combination thereof. Embodiments of the invention may also be implemented as instructions stored on a machine-readable medium, which may be read and executed by one or more processors. A machine-readable medium may include any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computing device). For example, a machine-readable medium may include read only memory (ROM); random access memory (RAM); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other forms of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.), and others.

[0012] An embodiment of a computing device is shown in FIG. 1. The computing device may comprise a processor 100 and a chipset 102 coupled to

one another via a processor bus 104. The processor 100 may comprise a memory controller 106 to dispatch read and/or write memory transactions on the processor bus 104. Moreover, the memory controller 106 of the processor 100 may provide the chipset 102 with page close messages. In one embodiment, the processor memory controller 106 may determine based upon pending memory transactions of the processor 100 whether closing open pages 108 of a memory 110 (See, FIG. 3) may be beneficial to memory performance. In response to determining that closing open pages 108 may be beneficial, the processor memory controller 106 may provide the chipset 102 with a page close message that requests the chipset 102 to close a page 108 of the memory 110.

[0013] The chipset 102 may comprise one or more integrated circuit packages or chips that couple the processor 100 to the memory 110 via a memory bus 112 and to other components 114 of the computing device such as, for example, BIOS firmware, keyboards, mice, storage devices, network interfaces, etc via one or more other buses 116. In particular, the chipset 102 may include a memory controller 118 to access the memory 110 via the memory bus 112 in response to memory transactions received from the processor 100 and other components 114 of the computing device. In one embodiment, the chipset memory controller 118 may determine based upon pending memory transactions received from the processor 100 and other components 114 whether closing open pages 108 of the memory 110 may be beneficial to memory performance. In response to determining that closing open pages 108 may be beneficial, the chipset memory

controller 118 may cause the memory 110 to close one or more open pages 108 of the memory 110.

[0014] Another embodiment of a computing device is shown in FIG. 2. The computing device of FIG. 2 may comprise a processor 120 and a chipset 102 coupled via a processor bus 104. The chipset 102 may comprise one or more integrated circuit packages or chips that couple the processor 100 to other components 114 of the computing device such as, for example, BIOS firmware, keyboards, mice, storage devices, network interfaces, etc.

[0015] The processor 100 may further comprise a memory controller 122 coupled to memory 110 via a memory bus 112. The memory controller 122 may access the memory 110 in response to memory transactions of the processor 100 and/or memory transactions received from the chipset 102 on the behalf of the other components 114 of the computing device. In one embodiment, the processor memory controller 106 may determine based upon pending memory transactions of the processor 100 and other components 114 whether closing open pages 108 of the memory 110 may be beneficial to memory performance. In response to determining that closing open pages 108 may be beneficial, the processor memory controller 122 may cause the memory 110 to close one or more open pages 108 of the memory 110.

[0016] The memory 110 of FIG. 1 and FIG. 2 may comprise various memory devices that provide addressable storage locations which the memory controllers 118, 122 may read data from and/or write data to. The memory 110 may comprise one or more different types of memory devices such as, for example,

DRAM (Dynamic Random Access Memory) devices, SDRAM (Synchronous DRAM) devices, DDR (Double Data Rate) SDRAM devices, or other memory devices. As shown in FIG. 3, the memory devices of the memory 110 may be arranged in a hierarchal manner. In particular, the memory 110 may comprise one or more banks 124. Each bank 124 may comprise one or more pages or rows 108. Further, each page 108 may comprise one or more columns 126. In other embodiments, the memory 110 may comprise more hierarchal levels than depicted in FIG. 3. Similarly, the memory 110 in other embodiments may comprise less hierarchal levels than depicted in FIG. 3.

[0017] In FIG. 4, there is shown an embodiment of a memory controller that comprises a page manager 200, a transaction queue 202, an address decoder 204, and a scheduler 206. In the computing device of FIG. 1, the processor memory controller 106 may comprise the circuitry of FIG. 4 in order to schedule memory transactions and provide the chipset 102 with page close messages for memory transactions dispatched to the chipset 102. Furthermore, the chipset memory controller 118 may also comprise the circuitry of FIG. 4 to schedule memory transactions to the memory 110 and to close pages 108 of the memory 110. Similarly, in the computing device of FIG. 2, the processor memory controller 122 may comprise the circuitry of FIG. 4 in order to schedule pending memory transactions and to close pages 108 of the memory 110.

[0018] The transaction queue 202 may store or queue memory transactions generated by the processor 100 in the case of the processor memory controllers 106, 122 or received from the processor 100 and/or other components 114 in

case of the chipset memory controller 118. In one embodiment, the transaction queue 202 may be implemented as a first-in-first-out (FIFO) queue having a tail into which memory transactions may be may pushed a head from which memory transactions may be popped. In another embodiment, the transaction queue 202 may be implemented via a plurality of storage elements that enable memory transactions to be removed from the transaction queue and dispatched toward the memory 110 in an order that differs from the order in which the memory transactions were stored in the transaction queue 202.

[0019] The address decoder 204 may decode addresses of the memory transactions stored in the transaction queue 202. The address decoder 204 may decode the addresses of the memory transactions to generate one or more selects that correspond to the hierarchal arrangement of the memory 110 and that may be used to select or address a particular storage location of the memory 110. In one embodiment, the address decoder 204 may determine and generate a bank select, a page select, and a column select that respectively select a bank 124, a page 108, and a column 126 of the memory 110. (See, FIG. 3.). In another embodiment, the address decoder 204 of the processor memory controller 106 may partially decode the addresses of the memory transactions. For example, the address decoder 204 of the processor memory controller 106 may only determine and generate a bank select and a page select for each memory transaction of the transaction queue 202.

[0020] The scheduler 206 may select memory transactions of the transaction queue 202 for processing based upon memory selects (e.g. bank, page, and/or

column selects) and associated page hit results of the memory transactions. In one embodiment, the scheduler 206 may schedule memory transactions of the transaction queue 202 such that the memory transactions are processed in an order that differs from the order in which the memory transactions were stored in the transaction queue 202. In particular, the scheduler 206 may attempt to increase memory performance by altering the processing order of the memory transactions stored in the transaction queue 202 in a manner that results in a greater number of page hits. However, it should be appreciated that the scheduler 206 may not be able to reorder memory transactions freely and still maintain data coherency. For example, data ordering rules of the memory controller 106, 118, 122 may prevent the scheduler 206 from moving a memory read transaction ahead of a memory write transaction if both transactions target a common storage location of the memory 110.

[0021] The memory controller 106, 118, 122 may further comprise a bus interface 208. In the case of the memory controllers 118, 122, the bus interface 208 may apply one or more selects such as, for example, the rank select, bank select, and page select to the memory bus 112 to open a page 108 of memory 110 associated with the memory transaction. The bus interface 208 may further apply one or more selects such as, for example, the column select to the memory bus 112 to select the column 126 of the open page 108 for reading and writing. Further, the bus interface 208 of the memory controllers 118, 122 may transmit a page close message such as, for example, a page close signal and/or transmit precharge command on the memory bus 112 to cause the memory 110 to close

the page 108. In the case of the memory controller 106, the bus interface 208 may generate one or more signals on the processor bus 104 to transmit a selected memory transaction and any corresponding data to the memory controller 118 of the chipset 102. Moreover, the bus interface 208 of the memory controller 106 may transmit a page close message such as, for example, a page close signal or precharge command on the processor bus 104 that indicates to the memory controller 118 of the chipset 102 whether to close the page 108 of the memory transaction.

[0022] The page manager 200 may adaptively determine whether to close pages 108 of the memory 110 based upon the effectiveness of previous page close messages. In one embodiment, the page manager 200 may comprise page state logic 210 and adaptive page close logic 212. The page state logic 210 may determine a page hit result for each memory transaction of the transaction queue 202 based upon memory selects for the memory transactions. The page state logic 210 may comprise bank registers 214 to track the most recently opened pages 108 of the banks 124 of the memory 110. In one embodiment, page state logic 210 may comprise a separate bank register 214 for each bank 124 of the memory 110 and each bank register 214 may store a page identifier such as, for example, a page select of the address decoder 204 that identifies the most recently opened page 108 of the bank 124. In particular, each bank 124 may comprise up to 2^A pages 108 and each bank register 214 may comprise at least A bits to store an A-bit page select that identifies one of the 2^A pages 108 of its respective bank 124.

[0023] The page state logic 210 may further comprise multiplexers 216 and comparators 218. In one embodiment, the page state logic 210 may comprise a separate multiplexer 216 and a separate comparator 218 for each memory transaction of the transaction queue 202. Each multiplexer 216 may select a bank register 214 based upon a bank select generated by the address decoder 204 for the respective transaction, and the comparator 218 may generate a page hit result for the respective memory transaction based upon whether the respective transaction targets the most recently opened page 108 of the bank 124. In one embodiment, the comparator 218 may generate a page hit result that indicates a page hit in response to determining that the page identifier of the selected bank register 214 is equal to a page select generated by the address decoder 204 for the respective memory transaction.

[0024] As depicted, the adaptive page close logic 212 may comprise a bank state register 220 may track actions performed upon the banks 124 of the memory 110. In one embodiment, the bank state register 220 may comprise a separate state bit or field for each bank 124 of the memory 110. Further, each state field 222 may store a bank state that may indicate among other things whether the adaptive page close logic 212 decided to close a page 108 of the bank 124 in response to a previous memory transaction to the bank 124. In one embodiment, the adaptive page close logic 212 may store a 1 in a state field 222 when the adaptive page close logic 212 decides to not close a page 108 of the respective bank 124, but may store a 0 in a state field 222 when the adaptive page close logic 212 decides to close a page 108 of the respective bank 124.

[0025] The page close logic 212 may adaptively generate page close messages based upon the page hit results of the page state logic 210, bank states of the bank state register 220, and bank selects of the address decoder 204. Further, in the case of the chipset memory controller 118, the adaptive page close logic 212 may generate page close messages based further upon page close messages received from the processor 100. The adaptive page close logic 212 may comprise circuitry, firmware, software, etc. that attempts to improve perceived memory performance by successfully predicting whether to close pages 108 of the memory 110 or leave them open. In general, the adaptive page close logic 212 may attempt to close pages 108 of the memory 110 in such a manner that attempts to increase the number of page-hit accesses and/or that attempts to decrease the number of page-miss accesses.

[0026] A page-hit access may occur in response to a memory transaction accessing a page 108 that was left open in response to a previous memory transaction. For a page-hit access, the memory controller 118 may leave the page 108 open after accessing a column 126 of the page 108 for the previous memory transaction and may access a different column 126 of the open page 108 for the current memory transaction. A page-miss access may occur in response to a memory transaction accessing a closed page 108 of a bank 124 that has another page 108 opened. For a page-miss access, the memory controller 118 may close the open page 108 of the bank 124, open another page 108 of the bank 124, and access a column 126 of the newly opened page 108 for the current memory transaction. A page-miss access generally has about three

times the latency as a page-hit access. A page-empty access may occur in response to a memory transaction accessing a closed page 108 of a bank 124 that has no pages 108 that are opened. For a page-empty access, the memory controller 118 may open a closed page 108 of the bank 124 and access a column 126 of the newly opened page 108 for the memory transaction. A page-empty access generally has about twice the latency as a page-hit access. Accordingly, the algorithms 224 may improve perceived memory performance by generating page close messages that increase the number of page-hit accesses and/or reduce the number of page-miss accesses to the memory 110.

[0027] In one embodiment, the adaptive page close logic 212 may comprise a neural network, genetic algorithm, and/or some other trainable circuit, firmware, and/or software logic that adapts to the effectiveness of previous page close determinations. In another embodiment, the adaptive page close logic 212 may comprise a plurality of algorithms 224 to make page close determinations and an algorithm selector 226 to select one of the plurality of algorithms 224 based upon the effectiveness of previous page close determinations.

[0028] In one embodiment, the adaptive page close logic 212 may determine that a previous page close determination was successful if the bank 124 is in an appropriate state for the current memory transaction. In particular, the adaptive page close logic 212 may determine that a previous page close determination was successful if the current memory transaction is a page-hit access and the previous page close determination for the bank 124 was to leave the page 108 open. Similarly, the adaptive page close logic 212 may determine that a previous

page close determination was successful if the current memory transaction is a page-miss access and the previous page close determination for the bank 124 was to close the page 108. On the other hand, the adaptive page close logic 212 may determine that a previous page close determination was unsuccessful if the current memory transaction is a page-hit access and the previous page close determination for the bank 124 was to close the page 108. Further, the adaptive page close logic 212 may determine that a previous page close determination was unsuccessful if the current memory transaction is a page-miss access and the previous page close determination for the bank 124 was to leave the page 108 open.

[0029] Referring now to FIG. 5, the algorithm selector 226 in one particular embodiment may comprise a counter 228 such as, for example, a four-bit counter to track the effectiveness of previously page close messages. The algorithm selector 226 may further select between a first algorithm ALG_0 and a second algorithm ALG_1 based upon the count of the counter 228. In one embodiment, the first algorithm ALG_0 may close pages 108 more conservatively than the second algorithm ALG_1, and the algorithm selector 226 may update the counter 228 per the following TABLE 1.

[0030]

Current Transaction Hit/Miss	Page Close/Open	Increment/Decrement Decision
Hit	Open	No Action
Hit	Closed	Decrement

Miss	Open	Increment
Miss	Closed	No Action

TABLE 1

[0031] As a result of updating the count of the counter 228 per TABLE 1, a value of 1 for the most significant bit of the count may indicate more pages 108 have been left open that should have been closed than pages 108 have been closed that should have been left open. Therefore, an algorithm ALG_1 that aggressively closes pages 108 may be more appropriate when the most significant bit has a value of 1 than an algorithm ALG_0 that more conservatively closes pages 108. Conversely, a value of 0 for the most significant bit of the count may indicate that more pages 108 have been closed that should have been left open than pages 108 have been left open that should have been closed. Therefore, when the most significant bit of the count has a value of 0, the algorithm ALG_0 that conservatively closes pages 108 may be more appropriate than the algorithm ALG_1 that more aggressively closes pages 108.

[0032] Accordingly, the algorithm selector 226 in one embodiment may select the more conservative algorithm ALG_0 that is less likely to generate a page close determination to close a page than the more aggressive algorithm ALG_1. Further, the algorithm may continue to select and use the page close determination of the more conservative algorithm as long as the most significant bit of the counter 228 remains a 0. Conversely, the algorithm selector 226 may

select the more aggressive algorithm ALG_1 and may use its page close determination as long as the most significant bit of the count remains a 1.

[0033] In another embodiment, the algorithm selector 226 may continue to select the more conservative algorithm ALG_0 and its page close determination until a first threshold THR_0 is passed. Similarly, the algorithm selector 226 may continue to select the more aggressive algorithm ALG_1 until a second threshold THR_1 is passed. Using separate thresholds THR_0, THR_1 may prevent or reduce thrashing situations in which the algorithm selector 226 keeps switching between algorithms in a manner that results in consistently generating unsuccessful page close messages.

[0034] As indicated above, the first algorithm ALG_0 may implement a more aggressive page closing technique and the second algorithm ALG_1 may implement a more conservative page closing technique. In particular, the first algorithm ALG_0 may determine to close a page 108 if the page states and bank states of the page state logic 210 in combination with the bank selects of the address decoder 204 indicate that the transaction queue 202 contains no memory transactions to the same page 108 as the memory transaction selected by the scheduler 206 and contains at least one memory transaction to the same bank 124 as the selected memory transaction. Further, the first algorithm ALG_0 may determine to leave the page 108 open if the page states and bank states of the page state logic 210 in combination with the bank selects of the address decoder 204 indicate that the transaction queue 202 contains at least one

memory transaction to the same page 108 as the memory transaction selected by the scheduler 206.

[0035] On the other hand, the second algorithm ALG_1 may determine to close a page 108 if the page states and bank states of the page state logic 210 in combination with the bank selects of the address decoder 204 indicate that the transaction queue 202 contains no memory transactions to the same page 108 as the memory transaction selected by the scheduler 206. Further, the second algorithm ALG_1 may determine to leave the page 108 open if the page states and bank states of the page state logic 210 in combination with the bank selects of the address decoder 204 indicate that the transaction queue 202 contains at least one memory transaction to the same page 108 as the memory transaction selected by the scheduler 206.

[0036] Shown in FIG. 6 is an embodiment of a method that may be used by the memory controllers 106, 118, 122 to adaptively generate page close messages and to close pages 108 based upon such page close messages. In block 300, the address decoder 204 may decode or partially decode addresses of memory transactions queued in the transaction queue 202. In one embodiment, the address decoder 204 may decode or partially decode physical addresses of the queued memory transactions to obtain bank selects and page selects for queued memory transactions. The computing device in some embodiments may support virtual addressing in which the processor 100 uses virtual addresses to access storage locations of the memory 110. The memory controller 106, 118 or 122 may translate or otherwise map the virtual addresses

to physical addresses that are ultimately decoded to access storage locations of the memory 110. In such virtual addressing embodiments, the address decoder 204 may decode or partially decode virtual addresses instead to obtain virtual bank selects and virtual page selects.

[0037] In response to the receiving bank selects and page selects for the queued memory transactions, page status logic 210 in block 302 may provide adaptive page close logic 212 with page hit results for the queued memory transactions. In block 304, the scheduler 206 may select a memory transaction of the transaction queue 202 and may provide the adaptive page close logic 212 with an transaction identifier that identifies the selected memory transaction. In one embodiment, the scheduler 206 may select the memory transaction based upon, for example, the memory selects of the queued memory transactions, the page hit results of the page status logic 210, the bank states of the bank state register 220, and/or data coherency ordering rules of the scheduler 206.

[0038] While the processor 100 may provide chipset 102 with page close messages, the memory controller 118 may elect to either close or open the page 108 based upon criteria in addition to the page close message received from the processor 100. For example, the memory controller 118 may elect to close the page 108 and process another memory transaction from another processor (not shown) or component 114. Further, the memory controller 118 and or memory 110 may elect to close the page 108 and dynamically refresh the page 108 or other pages 108 of the memory 110 between the memory transaction and the second memory transaction.

[0039] In block 306, the adaptive page close logic 212 may adapt its generation of page close messages based upon the page hit result and bank state associated with the selected memory transaction. Further, the adaptive page close logic 212 may generate a page close message that indicates whether the memory 110 is to close the page 108 of the selected memory transaction. In one embodiment, the algorithm selector 224 may update the counter 228 per above TABLE 1 and may switch between page close determinations of the algorithms ALG_0 and ALG_1 based upon the count passing the thresholds THR_0 and THR_1. Furthermore, the algorithm selector 226 may update the state field 222 of the bank state register 220 that corresponds to the bank 124 to be accessed by the memory transaction.

[0040] The bus interface 208 in block 308 may dispatch the memory transaction identified by the scheduler 206 and the page close message generated by the adaptive page close logic 212. In the case of the processor memory controller 106 of FIG. 1, the bus interface 208 may transfer the memory transaction and its associated page close message to the chipset 102 via the processor bus 104 for further processing by the chipset memory controller 118. In the case of the chipset memory controller 118 of FIG. 1 and the processor memory controller 122 of FIG. 2, the bus interface 208 may generate memory select signals (e.g. bank select, page select, column select) signals and other memory control signals (e.g. write enables, row address strobes, column address strobes) on the memory bus 112 to transfer data per the selected memory transaction. Moreover, if the associated page close message requests a page

108 be closed, the bus interface 208 may provide the memory 110 with a precharge signal, a precharge command, and/or some other page close signal or message that causes the memory 110 to close the page 108 accessed by the memory transaction.

[0041] Certain features of the invention have been described with reference to example embodiments. However, the description is not intended to be construed in a limiting sense. Various modifications of the example embodiments, as well as other embodiments of the invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.